REMARKS

Claims 1, 3 to 14, and 16 to 28 are pending in this application. Of these, claims 1 and 14 are independent.[1] Favorable reconsideration and further examination are respectfully requested.

All of the claims continue to stand rejected over (i) U.S. Patent Publication No. 2004/0126750 (Theilmann) in view of U.S. Patent Publication No. 2004/0002049 (Beavers), and (ii) U.S. Patent Publication No. 2002/0168621 (Cook) in view of U.S. Patent Publication No. 2006/0059253 (Goodman).

Regarding the rejection over Theilmann and Beavers, we still submit that the claims define over those references. However, we also note that Theilmann is not prior art to the subject application and, therefore, the rejection should be withdrawn.

More specifically, under 35 U.S.C. §103(c)(1): Subject matter developed by another person, which qualifies as prior art only under one or more of subsections (e), (f), and (g) of section 102 of this title [35 USC 102], shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the claimed invention was made, owned by the same person or subject to an obligation of assignment to the same person.

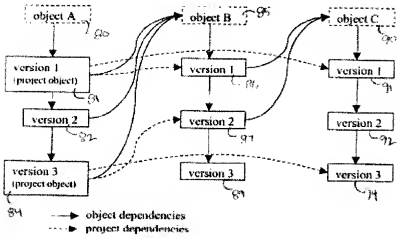
Both Theilmann and this application are assigned to SAP AKTIENGESELLSCHAFT, as evidenced by the assignment information for both applications printed from PAIR and attached hereto. Accordingly, withdrawal of the rejection over Theilmann is respectfully requested.

Independent claim 1 has been amended to recite that the version dependency data identifies versions of other learning objects upon which the project object depends, and to recite

[1] The Examiner is urged to independently confirm this recitation of the pending claims.

that the other learning objects, including the first and second objects, do not store version dependency data. The other learning objects store dependency data that identifies an object dependency but that does not identify a version dependency, and the other learning objects rely on the version dependency data in the project object for identification of version dependency. While it is believed that the former claims define over the art, these amendments have been made in an attempt to clarify what we mean by the different terms "**dependency data**" and "**version dependency data**", and thereby clarify any issues for appeal.

An embodiment of claim 1 is shown in Fig. 8 of the application, which is reproduced below.



As explained in the previous response, a project object 81 stores version dependency data, which is identified by the dashed lines. The version dependency data identifies the versions of other objects, here object 85 and object 90, upon which the project object 81 depends. In this example, object 85 exists in three versions: 86, 87 and 89; and object 90 exists in three versions: 91, 92

94. The version dependency data identifies the version of the object upon which the project object depends. A first object, e.g., object 85, contains dependency data, which is identified by solid lines. The dependency data identifies a second object, e.g., object 90, upon which the first object 85 depends, but does not identify the version upon which object 90 depends. The version dependency data is stored in the project object. The other learning objects therefore rely on the project object for identification of version dependency.

Cook fails to disclose or to suggest the foregoing features of claim 1. In this regard, it was admitted on pages 15 and 17 of the Office Action that Cook does not disclose version dependency data. Goodman was again cited to make up for this deficiency. In particular, the following paragraphs of Goodman were again cited for their alleged disclosure of object and data dependencies: 0138, 0170, 0173, 0316 and 0378.

As explained in paragraph 0176, Goodman describes a system that keeps track of different versions of software, e.g., through version control services, as described below.

> As with standard development code, when media content is created and edited, the version control services maintain a revision history of changes. This way, if it is necessary to revert to an original piece of media content, it is not necessary to go all the way back to the original source (which in the case of finding an image in a CD-ROM library containing 10,000 images, for example, could be a difficult task). In practice, this may mean storing the original and final copies of media (especially where volume is an issue). For this reason, a process for managing multiple versions of media content is put into place in the preferred development architecture 500.[2]

The Office Action cites the following paragraphs 0316 and 0378 (below) of Goodman for allegedly describing storage of version dependency data in a project object and dependency data, but not version dependency data, in objects that depend from the project object.

[2] Paragraph 0176

> [0316] As illustrated in FIG. 26, the information management tools 602 is also responsible for object management 646. Object management tools provide capabilities for viewing objects, their methods and attributes, and the dependencies between these objects. Object management tools also provide specific analysis tools, in order to understand interdependencies between the core classes and the components. When classes and components are modified, impact analysis tools are required to see where the modified entity is being used, allowing them to understand what is the overall impact of the change. This is more complex than with traditional systems as a veritable spider's web of dependencies between classes, components, and applications may ensue. In addition, OO features such as inheritance and polymorphism make tracking down dependencies with simple text search tools much more difficult.
>
> [0378] Process modeling tools 600 provide a graphical depiction of the business functions and processes being supported by a system. The tool(s) selected must support the modeling techniques being used in the development methodology; these include process decomposition, data flow and process dependency. Process modeling is a method for clarifying and communicating the business design of the system. The process model can provide an effective means of bringing people together, creating a shared vision of how the business is to function. A process model provides a means of standardizing a set of similar processes, which exist, for example, at different branches of the business.

As explained previously, paragraph 0316 does describe dependencies among objects, and "specific analysis tools...to understand interdependencies between the core classes and the components". Paragraph 0378, on the other hand, merely mentions "process dependency". Significantly, however, neither these paragraphs, nor the remainder of Goodman, describes exactly how the dependencies are analyzed or managed in connection with different versions.

While we agree that Goodman does describe version dependency, we do not understand Goodman to disclose or to suggest the claimed method. For example, nowhere does Goodman disclose or suggest two types of data: version dependency data and dependency data. Nor does Goodman disclose or suggest that one type of data - the version dependency data - is stored in a project object but not in other objects upon which the project object depends.

Paragraphs 0138, 0170 and 0173 (below) of Goodman were once again cited for their alleged disclosure of version control and storing dependency data.

[0138] The environment management process services 560 supports the environment where management processes are performed, and where systems are being built. The release management process services 562 manage the simultaneous development of multiple releases. The configuration management process services 564, often closely linked with release management, cover the version control, migration control, and change control of system components, such as code and its associated documentation. The problem management process services pertain to the problem tracking and solution process.

[0170] Another important distinction maintained by the folder management services 572 is the one between work in progress and completed documents that have been approved. This distinction is supported by a folder structure with carefully chosen access rights. An example of how the folder management services 572 maintain documentation for an application is illustrated in FIG. 21. As illustrated, documentation is divided into two general categories, work-in-progress and completed documentation. Work-in-progress can further be divided into teams, wherein each team maintains its own technical documentation and user documentation. Completed documentation is broken down by versions; including a current version folder and a previous version folder that contain technical documentation and user documentation.

[0173] The three major processes that are required to support media content management are: storage management services; metadata management services; and version control services. Storage management services deal with the methods of storing and retrieving media content. The cost of data storage may be decreasing, but it is still the case that for large volumes of media it is often uneconomical to store everything on-line. For this reason, processes are implemented with the storage management services that manage where data is stored, and how it is transitioned from one location to another within the netcentric computing system 10. There are three ways data is stored in the present invention: on-line (Instant access, for example, hard disk); near-line (delayed access, for example, CD-ROM jukebox); and, off-line (manual access, for example, CDs or tapes on shelves).

The foregoing paragraphs were once again cited without any explanation. Paragraph 0138 does mention version control, which is admittedly disclosed in Goodman. However, the *claimed method* of performing version control is very clearly not disclosed or suggested.

In response to our last Amendment, the Examiner stated the following:

Applicant argues that there is no teaching in Cook alone or in combination with Goodman of storing version dependency data in the project object where the version dependency data identifies at least a version of object.

In response to Applicant's argument, the Examiner submits that in the previous office action and rejection examiner stated that Cook alone did not teach the version dependency data as claimed. Instead Goodman taught the version dependency data in paragraphs 0316 and 0378. Goodman furthermore teaches version control in paragraphs 0138, 0170 and 0173.

We respectfully submit that our arguments in the last Amendment were not only that "Cook did not teach version dependency data". In fact, we acknowledged, in our response, that the Examiner admitted that Cook does not disclose version dependency data.[3] The crux of our arguments were (and continue to be) that Goodman does not make up for the glaring deficiencies of Cook. In fact, in our last Amendment, as we do again here, we laid out each paragraph relied upon by the Examiner, and explained how each paragraph does not show the features for which it was cited. There was absolutely no response, in the Office Action, to our arguments regarding Goodman, which we believe were clear and convincing. To the extent there was even any acknowledgement, it was simply to add the new claim features to the existing rejections in the Office Action, leaving all the same citations exactly has they had been.

In this regard, we respectfully direct the Examiner's attention to MPEP §707.07(f), which reads:

> Where the applicant traverses any rejection, the examiner should, if he or she repeats the rejection, take note of the applicant's argument and answer the substance of it.

[3] See Amendment of November 8, 2007, page 13

We respectfully submit that our arguments over Goodman were not answered. Should the Examiner repeat the rejection over Goodman, we respectfully request that the Examiner explain why Goodman makes up for the deficiencies of Cook. That is, we know Cook does not show dependency data (as admitted by the Examiner). Then, the questions becomes whether Goodman shows dependency data *of the type claimed*, not just dependency data in general. In other words, does Goodman show both dependency data and version dependency data (which, as claimed, are two different things)? We assert that it does not. Therefore, Goodman could not possibly disclose or suggest the method claimed, even when combined with Cook.

For at least the foregoing reasons, we submit that even if Goodman were combined with Cook in the manner suggested, the resulting hypothetical combination would fail to disclose or to suggest the features of claim 1.

Amended independent claim 14 is a computer program product claim that roughly corresponds to independent claim 1, and is also believed to be patentable.

The dependent claims are also believed to define patentable features of the invention. Each dependent claim partakes of the novelty of its corresponding independent claim and, as such, each has not been discussed specifically herein.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

In view of the foregoing amendments and remarks, we respectfully submit that the application is in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

The undersigned attorney can be reached at the address shown below. All telephone calls should be directed to the undersigned at 617-521-7896.

Please apply any fees or credits due in this case to Deposit Account 06-1050 referencing Attorney Docket No. 13909-161001.

Respectfully submitted,

Date: May 13, 2008

[illegible]
Paul A. Pysher
Reg. No. 40,780

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21921206.doc